МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №6
по дисциплине «Искусственные нейронные сети»
Тема: «Прогноз успеха фильмов по обзорам»

Студент гр. 7381 _____ Лукашев Р.С.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель работы.

Прогноз успеха фильмов по обзорам (Predict Sentiment From Movie Reviews)

Задачи.

- Ознакомиться с задачей регрессии
- Изучить способы представления текста для передачи в ИНС
- Достигнуть точность прогноза не менее 95%

Требования.

1. Построить и обучить нейронную сеть для обработки текста
2. Исследовать результаты при различном размере вектора представления текста
3. Написать функцию, которая позволяет ввести пользовательский текст (в отчете привести пример работы сети на пользовательском тексте)

Ход работы.

Была найдена архитектура ИНС, дающая точность ~89% (см. рис. 1)

```
model.add(Dense(50, activation="relu", input_shape=(num_words,)))
model.add(Dropout(0.5, noise_shape=None, seed=None))
model.add(Dense(30, activation="relu"))
model.add(Dropout(0.2, noise_shape=None, seed=None))
model.add(Dense(10, activation="relu"))
model.add(Dense(1, activation="sigmoid"))
```

Рисунок 1 – выбранная архитектура модели.

Вообще, практически любая архитектура, состоящая из Dense и Dropout слоев, дает приблизительно одинаковую ~89% точность, меняется разве что время и скорость обучения. При увеличении количества эпох неизбежно проявляется переобучение.

Далее исследуем результаты при различном размере вектора представления текста (см. рис. 2)

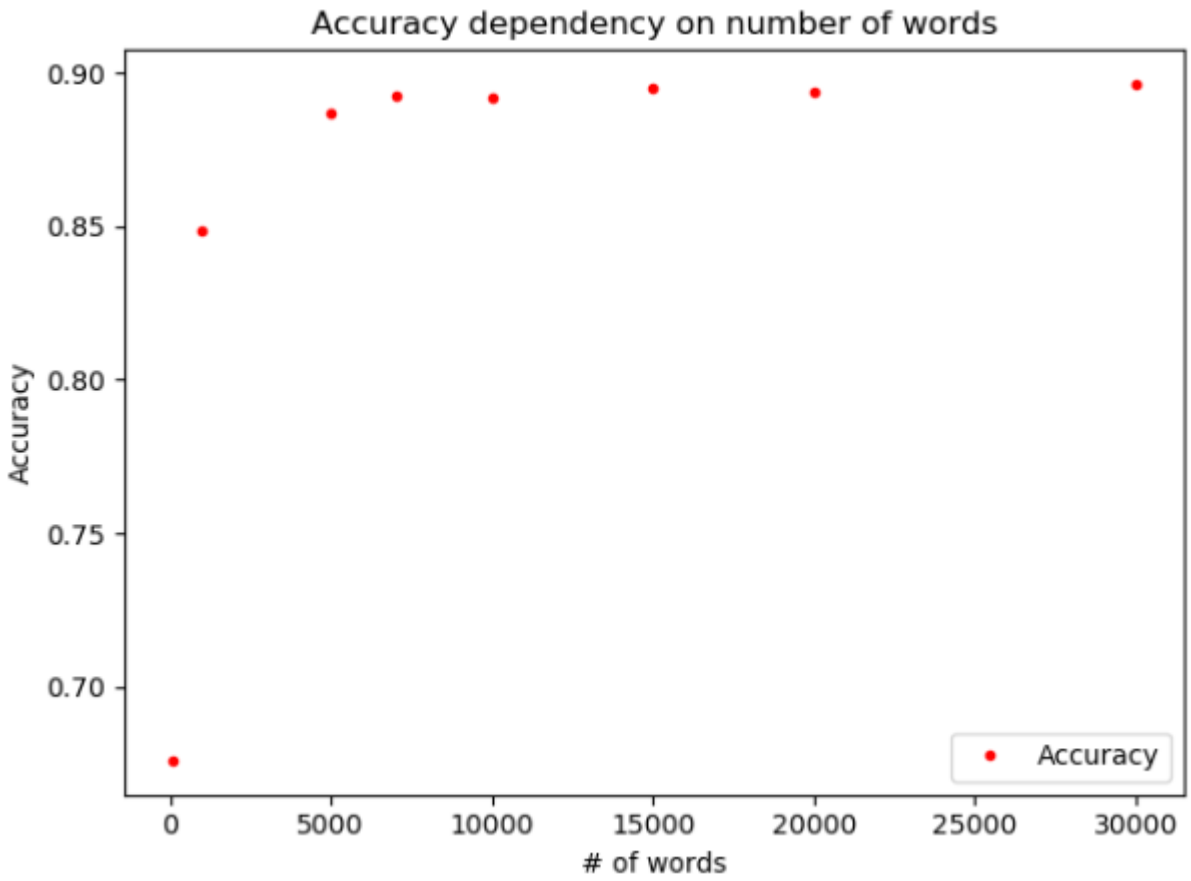

Рисунок 2 – зависимость точности модели от размера вектора представления текста.

Как можно заметить, чем больше длина, тем точнее модель, но тем медленнее она обучается.

Была написана функция, позволяющая обрабатывать пользовательский текст (см. рис. 3).

```python
def predict_user_text(filename):
    f = open(filename, 'r')
    open_text = f.read()
    f.close()
    table = str.maketrans('', '', string.punctuation)
    open_text = [w.translate(table) for w in open_text.lower().split()]
    indices = imdb.get_word_index()
    text = []
    for i in open_text:
        if i in indices and indices[i] < num_words:
            text.append(indices[i])

    text = vectorize([text], num_words)

    result = model.predict_classes(text)
    print(filename)
    if result == 1:
        print('Положительный отзыв')
    else:
        print('Отрицательный отзыв')
```

Рисунок 3 – функция, позволяющая обрабатывать пользовательский текст.

На рисунках 4, 5 приведен пример пользовательского текста и результата работы ИНС на этом пользовательском тексте.

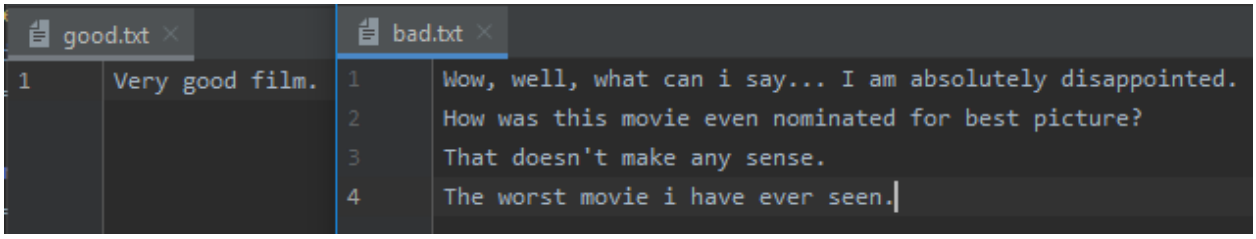

Рисунок 4 – пользовательский текст.

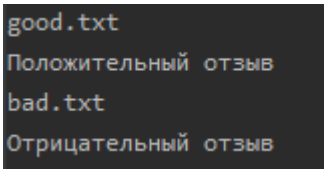

Рисунок 5 – результат работы ИНС на пользовательском тексте.

Выводы.

В ходе выполнения данной лабораторной работы была изучена задача класификации по настроению обзоров из датасета IMDB. Были изучены способы представления данных для работы с ИНС, и исследованы результаты при различном размере вектора представления текста в частотном виде.

Была обучена модель, которая с точностью 89% определяет настроение обзора, а также написана функция, позволяющая определять настрой пользовательского обзора.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import string
import matplotlib.pyplot as plt
import numpy as np
from keras.datasets import imdb
from keras.layers import Dense, Dropout
from keras.models import Sequential


def vectorize(sequences, dimension):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results


def predict_user_text(filename):
    f = open(filename, 'r')
    open_text = f.read()
    f.close()
    table = str.maketrans('', '', string.punctuation)
    open_text = [w.translate(table) for w in open_text.lower().split()]
    indices = imdb.get_word_index()
    text = []
    for i in open_text:
        if i in indices and indices[i] < num_words:
            text.append(indices[i])

    text = vectorize([text], num_words)

    result = model.predict_classes(text)
    print(filename)
    if result == 1:
        print('Положительный отзыв')
    else:
        print('Отрицательный отзыв')


nw = [100, 1000, 5000, 7000, 10000, 15000, 20000, 30000]
ac = []
for num_words in nw:
    (training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=num_words)
    data = np.concatenate((training_data, testing_data), axis=0)
    targets = np.concatenate((training_targets, testing_targets), axis=0)

    data = vectorize(data, num_words)
    targets = np.array(targets).astype("float32")

    test_x = data[:10000]
    test_y = targets[:10000]
    train_x = data[10000:]
    train_y = targets[10000:]

    model = Sequential()

    model.add(Dense(50, activation="relu", input_shape=(num_words,)))
    model.add(Dropout(0.5, noise_shape=None, seed=None))
```

```
    model.add(Dense(30, activation="relu"))
    model.add(Dropout(0.2, noise_shape=None, seed=None))
    model.add(Dense(10, activation="relu"))
    model.add(Dense(1, activation="sigmoid"))

    model.compile(optimizer="adam",
                  loss="binary_crossentropy",
                  metrics=["accuracy"])

    results = model.fit(train_x, train_y,
                        epochs=2, batch_size=500,
                        validation_data=(test_x, test_y))

    ac.append(np.mean(results.history["val_accuracy"]))

plt.plot(nw, ac, 'r.', label='Accuracy')
plt.title('Accuracy dependency on number of words')
plt.xlabel('# of words')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

predict_user_text("good.txt")
predict_user_text("bad.txt")
```